

EXPRESS MAIL LABEL NO.: ET9443273505

DATE OF DEPOSIT: Jan. 10, 2002

I hereby certify that this paper and fee are being deposited with the United States Postal Service Express Mail Post Office to Addressee service under 37 CFR §1.10 on the date indicated above and is addressed to the Assistant Commissioner of Patents, Washington, D.C. 20231.

Linda Dupont

NAME OF PERSON MAILING PAPER AND FEE

Linda Dupont

SIGNATURE OF PERSON MAILING PAPER AND FEE

INVENTOR: Erick M. Griffin

## Using Introspection for Access of Class Resource Bundle

### Information for Messages

### BACKGROUND OF THE INVENTION

#### Field of the Invention

5 The present invention relates to computer software, and deals more particularly with methods, systems, and computer program products for using introspection for access to class resource bundle information for messages.

#### Reservation of Copyright

A portion of the disclosure of this patent document contains material to which a claim of

copyright protection is made. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but reserves all other rights whatsoever.

### **Description of the Related Art**

5 Much computer software today contains interfaces between the functions of computer software and the user who interacts with those functions. The software will present information to the user in the form of menus or explanatory text with which the user will interact. In the case of menus, the user will make a selection and the computer software will decode the selection and perform the indicated operation. In the case of explanatory text, the user will indicate which  
10 portion of the explanatory text the user desires to be presented, by making a selection either from a table of contents, a list, or an outline; the computer software will respond with a block of text that it has stored within the program or on disk.

It often happens that the explanatory text is volatile and changes on a periodic basis. This necessitates a change in either the selection criteria (i.e. the text comprising the selection) or the  
15 textual block, both of which may either be stored on disk or within the program. Such changes may create the need to alter the computer software used to present the text. For example, in days when storage structures were static, an increase in the size of a particular textual block could require a corresponding increase in the space allocated within the computer program to store such text, since only a fixed amount of space was allocated based upon the maximum expected size of  
20 the textual block. Such an increase would require rebuilding the executable software module to

provide more storage. However, this required the executable software to be sized for the worst case and was therefore an inefficient use of available resources for storage and execution of the module.

Later as computer systems became more sophisticated, dynamic storage capabilities were developed, in which the computer software would determine the required amount of storage, based upon the size of the textual block, and then request that amount of storage from the computer system, which would then allocate an amount of space from its pool of available space. While this was an efficient management of space among numerous computer software programs having different space requirements, there were a number of problems associated with dynamic storage. First, each computer software program running under such a computer system required the development of storage management schemes for acquiring and releasing storage blocks, which in turn resulted in longer processing time required for the computer software module. This required careful attention by the programmer to ensure that memory was being properly managed. Second, such schemes were known to be prone to problems, such as “memory leaks” in which, through error by either the programmer or system timing problems, a storage block was thought to have been released but in reality was never recognized as being released by the computer system. This resulted in a gradual buildup of unreleased memory blocks that were not allocated to any executing program, and this reduced the amount of available memory for allocation by the system. The computer system gradually ground to a halt through lack of resources and frequently had to be reinitiated. Other problems consisted of erroneous computer software “stepping on” the memory of other computer software by impermissibly writing into its memory. Third, because

each computer software module was required to provide its own memory management scheme consisting of requesting, releasing, and keeping track of memory allocations, the result was replication of code among many different software modules.

One solution to the problem of duplicate memory management schemes within different software programs using the same textual data was to store the data on a disk file which was commonly accessed by each software module. In this way, text could be expanded or contracted without affecting the computer software accessing it. However, the organization of the data within the disk file had to be known to the computer software which would then implement a method for accessing this dynamic data on disk. This scheme of accessing and processing dynamic disk data had the same problem mentioned previously of replication of the same or similar computer code within different computer programs, resulting in duplicated software maintenance effort on the part of the programmers and inefficient use of computer resources (in terms of software module sizes). This problem was ameliorated somewhat by the development of software libraries which, having the common computer code, would then be included in each computer software program which accessed the particular disk files; but whenever this library code changed, it necessitated the rebuilding of each computer program using the library. Furthermore, dependencies within the library had to be recognized and provided for within each computer program. Finally, a computer software program might be omitted from the rebuild process, either inadvertently or because the library change was not judged to be relevant to the computer software program; this resulted in many different software programs having many different configurations.

Another solution to the problem of duplication of memory management schemes by different software programs accessing the same textual data was to store the data within an executable service module which was then dynamically linked to the program when it was invoked at runtime. This latter solution resulted in the reduction in importance of system libraries since the routines for accessing and providing the textual data could be built into the executable service module and used by all software programs. However, the programs still had to be aware of the nature of the data and such dependencies were a part of each computer program. Each program had limits to its functionality; if the textual data within the executable service module changed too much, in terms of its content, interpretation, etc., then the using computer program would still have to be modified to accommodate such changes.

Today, the concept of object-oriented design has been implemented within the software development environment; this approach has been described and documented in “Object-Oriented Analysis and Design with Applications”, 2nd Edition, by Grady Booch, et al. (Addison Wesley Longman, Inc., 1994, ISBN 0-8053-5340-2), which is incorporated herein by reference. The intent is to isolate, or encapsulate, properties and methods associated with the object of interest so that dependencies between the particular software program functions and the functions associated with the object are minimized. A number of computer languages have been developed to implement these object-oriented design concepts, such as Ada, C++, and Java, and each language has its own strengths, capabilities, and weaknesses as to how well it reduces unwanted dependencies within programmatic structures. (“Java” is a trademark of Sun Microsystems, Inc.)

In particular, the Java computer language is both a programming language in its own right and a platform for program development. Programs written in Java are usually implemented in practice as a two stage process, in which the Java source code is compiled to create an intermediate code which is then interpreted by the Java platform implemented on the particular computer system on which the Java program is executed. This characteristic makes Java programs transportable between different computer systems. Programs written in Java can also be implemented as a completely compiled program running under, and using the services of, a particular computer system. The Java platform is comprised of the Java Virtual Machine (JVM), which serves as the system-dependent platform upon which is executed the system-independent Java program, and the Java Application Programming Interface (Java API), which provides a large set of commonly-used software components to the program. Different subsets of these components are grouped by similar function and are called libraries, or packages. For example, one subset of these components consists of the graphical user interface (GUI) containing generic objects called widgets used by the using program to interact with a user at a computer terminal. Another Java API library is called JavaBeans, which is a trademark of Sun Microsystems, Inc.

The term “reflection” describes a Java API which supports introspection of the classes and objects in the JVM. By using reflection, a program can dynamically examine an arbitrary object at run time and determine its characteristics, i.e. details about its class, interfaces it implements, its list of methods, and the fields it supports.

One particular Java API library provides a set of tools supporting internationalization of a

software application, i.e. simplifying the application's adaptation to various languages and regions without software changes. This package is called "java.util" and it includes the following three classes (within the meaning of object-oriented terminology): Locale, TimeZone, and ResourceBundle. The abstract class ResourceBundle has two subclasses:

5   PropertyResourceBundle and ListResourceBundle. A PropertyResourceBundle is backed by a properties file, a plain-text file that contains translatable text. Properties files are not part of the Java source code, and they can contain values for String objects only. The ListResourceBundle is backed by a class file for encapsulating locale-specific objects. ResourceBundle objects contain an array of key-value pairs for values which have meaning for a locale specified in a Locale object. To retrieve a value from the ResourceBundle, the corresponding key, which must be defined as a String type, must be presented to the ResourceBundle; the value is the locale-specific object, usually a String type also.

10       The problem addressed by internationalization is that of dynamically supplying special objects which are pertinent to the locale in which the program is being run, without changing the source code. However, the structures used to specifically address internationalization can also be used to address other, more common problems. One such problem is being able to supply a split pane help panel, where a list of topics is displayed in the left panel and corresponding help text for a selected subject or topic is displayed in the right panel. An internal data structure within the program component must be maintained to keep the left and right panels in synchronization; this data structure normally takes the form of an array or vector. Such an internal data structure and the source code which supports it must be maintained as part of normal software maintenance

procedures; this increases the cost of software maintenance and the opportunities for introducing errors during normal software maintenance activities. Furthermore, every time such code is changed, the possibility is introduced that an unwanted dependency is created which will provide hidden sources of error.

5           Accordingly, what is needed is an improved technique for accessing textual data information and maintaining it in an efficient, rapid manner without creating unwanted dependencies between the content, organization, mode of access of such data and the requesting program.

### **SUMMARY OF THE INVENTION**

10           An object of the present invention is to provide an improved technique to allow a program to access data without creating unwanted dependencies between the content, organization, mode of access of such data and the requesting program.

          Another object of the present invention is to provide a technique which is not dependent upon the search methodology used to locate the data.

15           Yet another object of the present invention is to provide a technique which provides textual data corresponding to a key without being dependent upon the storage organization of the textual data.



Still another object of the present invention is to provide a technique which provides textual data corresponding to a key without being dependent upon the storage organization of the key.

Yet another object of the present invention is to provide a technique which requires only the name of the object containing the required methods and required data, in order to use the object through introspection.

Other objects and advantages of the present invention will be set forth in part in the description and in the drawings which follow and, in part, will be obvious from the description or may be learned by practice of the invention.

To achieve the foregoing objects, and in accordance with the purpose of the invention as broadly described herein, the present invention provides methods, systems, and computer program products for allowing a program to access a particular data structure without creating unwanted dependencies between the requesting program and the content, organization, and mode of access of such data. The data to which the invention applies comprise a first set of objects each having an associated first key, a second set of objects each having an associated second key, and a one-to-one relationship between a first key associated with a first object and a second key associated with a second object. The data and their associated methods are encapsulated in a Java ListResourceBundle, and the characteristics of the data contained in the ListResourceBundle are ascertained by reflection/introspection of the data contained therein. In this way, dependencies

between the data and the using program are eliminated. In particular, the use of a  
ListResourceBundle requires that the keys be cast as textual strings. The ListResourceBundle  
contains two items -- the standard non-extendable two dimensional array structure normally found  
in a resource bundle and containing pairs of keys and corresponding values, and a special public  
5 two dimensional array structure containing key pairs of associated keys.

The special two dimensional array of key pairs mirrors a GUI presentation of two panels  
which are available for display of objects. It establishes a one-to-one relationship between an  
object taken from a first set of objects and an object taken from a second set of objects. A first  
panel displays a first set of objects, each identified by the first key of a selected key pair, and a  
10 second panel displays the corresponding object taken from a second set of objects and identified  
by the second key in the key pair. When an object is selected from the first panel, its  
corresponding object is found through the special two dimensional array and displayed in the  
second panel. The invention enables the sets of objects to be modified without corresponding  
modification of programmatic code; it also enables the dual panel protocol described above to be  
15 efficiently implemented. (Note that while the examples and discussions herein are primarily in  
terms of the first and second panels containing textual data, this is for purposes of illustration and  
not of limitation. Alternative types of information, such as representations of graphical icons,  
might be stored as values in the two dimensional array and presented on either or both sides of the  
display.)

20 The present invention will now be described with reference to the following drawings, in

which like reference numbers denote the same element throughout.

## **BRIEF DESCRIPTION OF THE DRAWINGS**

FIG. 1 is an example of a GUI split pane window having two panels, as seen by a user of a program which has been instrumented according to the invention;

FIG. 2 provides a diagram showing the internal data structure of an instantiation of a ListResourceBundle which has been structured according to the invention; and

FIG. 3 provides a relationship chart for a particular embodiment of the invention, giving the relationships between various Java methods employed to maintain a GUI screen.

## **DESCRIPTION OF PREFERRED EMBODIMENTS**

The present invention defines techniques for allowing a program, without creating unwanted dependencies, to access data of a type comprising a first set of objects each having an associated first key, a second set of objects each having an associated second key, and a one-to-one relationship between a first key and a second key, the keys associated with an object taken from the first set and an object taken from the second set, respectively. The first set of objects is preferably organized as a list (or similar structure) having an order imposed upon its components; in programming terms, this is commonly called an index. An example of such a list might be a table of contents of a manuscript in which each first object is a textual string representing a section heading. Each such topic would have a one-to-one logical relationship to explanatory text

which defines, describes, explains, or otherwise provides more information pertaining to the section heading. This explanatory text may be of any length, but as a unit will have a logical relationship with the section heading. If each section heading and each explanatory text block is associated with a unique key, then the relationship between the topic and its associated explanatory text block can be defined by a relationship between the two keys. Another example of such a list might be a list of help topics in which the first object is a textual string representing a particular topic and the second object is a block of explanatory text elaborating upon the particular topic.

To illustrate the inventive concept in a more concrete manner, it is instructive to examine an environment in which the invention has been advantageously employed. Such an environment is given by an internally-deployed tool referred to as the Test Case Result (TCR) Manager Tool, as will be described herein. TCR is part of an integrated test environment developed to support automated regression testing of previously developed and/or modified software to ensure consistent performance. A system user such as a test engineer uses the TCR application to query a database where data is stored, e.g. test case statistics or raw data being generated by a test harness. As part of the TCR, a Help capability (*TCRManagerHelp*) is provided to supply the user with assistance for a selected set of topics through display on a computer terminal of a split pane help window. This display is made through a standard GUI associated with Windows operating system. ("Windows" is a trademark of Microsoft Corporation.) Associated with each topic is a detailed explanation of the issues related to the topic, and the user is given the capability of searching through the available topics as by scrolling, selecting a topic through use of a mouse

click, and then observing the associated detailed explanation of the topic.

Referring now to FIG. 1, a graphical user interface (GUI) is shown illustrating an initial Help screen in which the invention may be used to produce a graphical display. A window **100** contains two panels. Left panel **110** is used to display a list portion **115** of a list of textual objects. In the example shown, the list is a table of contents where each textual object is a topic. Other list structures such as indented outlines, numbered lists, etc., might equally well be shown without departing from the scope of the invention. Such lists may also be either ordered or unordered. A particular list object **116** has been selected, which in an actual GUI display would typically be indicated by a highlighted area of the list. Right panel **120** is used to display explanatory text **125**, where the semantic content of text **125** corresponds to and elaborates on the selected list object **116**. A header area **117** in the right panel **120** contains a copy of the selected list object **116** and provides confirmation of the selection.

The display shown in FIG. 1 is presented and maintained by a computer program, which in preferred embodiments is a program written in Java and employing several Java API packages, such as the Java Swing API or the Java AWT API, both for GUI displays. Another Java package for internationalization defines the ListResourceBundle class from which a resource bundle object has been instantiated which implements the principles of the invention. Referring now to FIG. 2, a resource bundle called *TCRHelp* is illustrated, which is an extension of the standard Java ListResourceBundle class. In a preferred embodiment of the invention, this resource bundle contains two static string arrays. The first string array, *searchJListFields* **210**, is a public 2 x n

array with each entry containing two corresponding keys, a first key field **211** containing keys indexing search (or topic) text strings and a second key field **212** indexing help text blocks. In preferred embodiments, both keys are text strings within the meaning of Java (although other types of key values, such as integers, might be used alternatively). Both the search text string and the help text string are contained in a second array, *contents* **250**, which is also a static encapsulated 2 x n array. Each entry of the *contents* array **250** contains a key and either the corresponding help text information or the search/topic text, and both the key and help or the corresponding search text are implemented as text strings. Every key entry contained in the *contents* array **250** corresponds to an entry in the *searchJListFields* array **210**, and the *searchJListFields* array **210** is used to map the search elements to the actual message text elements. (As stated earlier, entries in the contents array are described herein as containing text, although this is simply for purposes of illustration. Other types of array values are within the scope of the present invention.)

In the example shown in FIG. 1, the topic string V1 having a value of "1.0 Introduction" (corresponding to selected list object **116** of FIG. 1) may have a key K1 having a value of "tcrmanagerhelp\_searchtext10". The text block V2 having a value of "Welcome to the Test Case ... than one test execution." (i.e. the corresponding help text shown at **125** of FIG. 1) may have a key K2 having a value of "tcrmanagerhelp\_helptext10". This illustrates the fact that the keys K1 and K2 are arbitrary and unique and have structures which are meaningful only to the program developer, whereas text strings V1 and V2 are representative of objects to be displayed on the computer terminal. It should be noted that this use of the resource bundle does not demonstrate

concepts of internationalization.

It should be appreciated by those skilled in the art of software programming that the structure of the *searchJListFields* array **210** and *contents* array **250** may be one of any number of other data structures used to relate two sets of objects besides two dimensional arrays, e.g. linked lists, hash lists, binary lists, binary trees, and other such structures well known to the art. Furthermore, the search method for traversing the structures and locating help information corresponding to a key may vary; such search methods as a binary search, linear search, random search, and the like could be used without departing from the scope of the invention. The use of the resource bundle in this manner isolates from the calling module the data, the data structure, and the search methods used to retrieve the data. Finally, the technique of the invention may also be applied to similar Java Swing components, such as JTree, without departing from the scope of the invention.

Referring now to FIG. 3, an embodiment **300** of a particular set of functions as implemented in Java code is given to illustrate the implementation of a help capability referred to herein as *TCRManagerHelp*. According to this implementation, two methods (or routines) are provided to create the GUI screen shown in FIG. 1, namely a SplitPane code for initializing and generating the pane shown in FIG. 1 and Resource Bundle code for providing the content of the split pane. Once the split pane window is initialized, then *TCRManagerHelp* becomes dormant until it is instructed to shut down the screen display and terminate. It is assumed that *TCRManagerHelp* is invoked from a user screen selection which causes *TCRManagerHelp* to be

activated; this code is not necessary for understanding the invention and is not given here.

When the user selects a Help command from other graphical displays (not shown), the *TCRManagerHelp* code **310** is invoked to create an initial Help screen **100**. This screen or window **100** is built up by creating the left panel **110** by calling the method *createSearchPanel()* **320** and then by creating the right panel **120** by calling the method *createTextPanel()* **330**. The *TCRManagerHelp* code **310** then returns. Subsequent actions such as *changeTextPanel()* **340** are preferably invoked by mouse listener **350**.

The *createSearchPanel()* method **320** preferably gets the resource bundle class name; introspects the public *searchJListFields* array; creates a JList pane in a scrolled panel, based on the initial array elements; and adds a mouse listener for mouse click events. Code which may be used to implement the *createSearchPanel()* method **320** is shown in Appendix A. Pertinent parts of this code are also given below, showing use of introspection to obtain the content of the *searchJListFields* array **210** in the resource bundle:

```
Class alt_rb = rb.getClass();  
Field stringArrayField = alt_rb.getField("searchJListFields");  
searchJListFields = (String[][]) stringArrayField.get(alt_rb);
```

The name of the resource bundle class and a reference to the actual class is obtained through a call to the *getClass()* method for resource bundles *rb*. The string array field for class reference



*alt\_rb* is obtained by using the *getField()* method. Once the field is obtained, the actual object of this field can be obtained using the *get()* method. When all this is completed, the local *searchJListFields* contains a reflected copy of the two dimensional array of values that are in the public resource bundle field.

5           The *createTextPanel()* method **330** preferably creates a text area for the subject header, and then creates a text area for the help body. Code which may be used to implement the *createTextPanel()* method **330** is shown in Appendix B.

10           The *changeTextPanel* method **340** is provided to make a change in the contents of second panel **120** based upon a selection in the first panel **110**, and is called by the mouse listener **350**. (Mouse listener **350** preferably operates by obtaining the index of the selected item; calling the *changeTextPanel()* method; recording the user action; and then repainting the window.) The method *changeTextPanel()* **340** preferably uses the index obtained by the mouse listener to replace the header (illustrated at **117** of FIG. 1), using the newly-selected field index and the introspected first item of the *searchJListFields* array, and then replaces the body of the text (illustrated at **125** of FIG. 1). Code which may be used to implement the *changeTextPanel()* method **340** is shown in Appendix C. Pertinent segments of this code will now be described in more detail.

The two dimensional array *searchJListFields* is used to map the search elements to the actual message text elements. When the user makes a selection of a search element, i.e. key text

string, within the left panel of the GUI interface, the two-dimensional array *searchJListFields* is used to retrieve the corresponding help text element for the right panel. Selected items are identified by an index into the object that is being used for the search. In the code provided, a Java *JList* is used, however any similar search object would suffice, such as, for example, a Java *JTree* search object. The new header value is obtained by the following line of code, based upon that index value:

```
String rb_tmp_str = rb.getString(searchJListFields[item_index][0]);
```

According to this Java code segment, a temporary string identified by *rb\_tmp\_str* is set with that value obtained by using a *getString()* method which is generic to resource bundles. The header area *TextField* is then modified according to the following line of code:

```
hdr_area.replaceRange (rb_tmp_str, 0, str_len);
```

According to this Java code segment, a generic resource bundle method *replaceRange()* is used to change the entire range of the header with the new header string. Similarly, the *TextField* for the help text message is changed using the same *replaceRange()* method, only using the second part of the array element currently indexed, according to the following code:

```
tmp=area.getText();  
str_len=tmp.length();  
area.replaceRange (rb.getString (searchJListFields[item_index][1], 0, str_len);
```

Referring again to the screen display shown in FIG. 1, when the mouse is clicked by a user to select a line in the first panel, the topic entry for the selection, the help message text, and header are automatically returned through the use of the introspected resource bundle array *searchJListFields* that had been obtained earlier. This array maps each search subject heading with its corresponding help message text. There are no individual *getString()* resource bundle message calls in the supporting code. Therefore, when the help capability implemented in the resource bundle is modified by adding new message headers and help text blocks therein, e.g. to add additional explanations, include additional topics, clarify existing help topics, provide internationalization, etc., then only the resource bundle is affected; no new code must be written or modified in order to accommodate the modified help capability. The code contained within the existing *TCRManagerHelp* class will dynamically pick up those changes after the resource bundle is rebuilt.

Appendix D provides sample code which may be used for handling user actions.

Appendix E provides sample resource bundle data that may be used to provide help text as illustrated with reference to the example in FIG. 1. (It should be noted that the present invention is not limited to use in providing help text.)

As will be appreciated by one of skill in the art, embodiments of the present invention may be provided as methods, systems, or computer program products. Accordingly, the present invention may take the form of an entirely hardware embodiment, an entirely software embodiment or an embodiment combining software and hardware aspects. Furthermore, the present invention may take the form of a computer program product which is embodied on one or more computer-usable storage media (including, but not limited to, disk storage, CD-ROM, optical storage, and so forth) having computer-usable program code embodied therein.

As has been demonstrated, the present invention provides advantageous techniques for allowing update of a textual object, such as a help dialog, without concurrent update of the calling program's code and for providing information about the textual object by default through introspection. By using few routines and introspection, most of the mapping that is needed to associate the help index with its text and header can be done through introspection. This is accomplished by defining a special mapping array within the resource bundle class and using introspection to load the appropriate help topic and text, as described above.

The present invention has been described with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems) and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer program instructions. These computer program

instructions may be provided to a processor of a general purpose computer, special purpose computer, embedded processor or other programmable data processing apparatus to produce a machine, such that the instructions, which execute via the processor of the computer or other programmable data processing apparatus, create means for implementing the functions specified in the flowchart and/or block diagram block or blocks.

These computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function specified in the flowchart and/or block diagram block or blocks.

The computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions specified in the flowchart and/or block diagram block or blocks.

While the preferred embodiments of the present invention have been described, additional variations and modifications in those embodiments may occur to those skilled in the art once they learn of the basic inventive concepts. Therefore, it is intended that the appended claims shall be

construed to include both the preferred embodiment and all such variations and modifications as fall within the spirit and scope of the invention.

RSW920010149US1

## APPENDIX A

```

/**
 * The <em>createSearchPanel</em> method creates a search
 * panel.
5  * <p>
 * @return void
 * (c) Copyright 2002 International Business Machines
 * Corporation
 */
10 private void createSearchPanel() {

    //
    // Create a model to hold our entries.
    //
    DefaultListModel model = new DefaultListModel();

15    //
    // Add elements to model from resource bundle.
    // We can do this by using reflection/introspection to get
    // the searchJListFields content in the
    // resource bundle and then populate our model from that.
20    //
    try {
        Class alt_rb = rb.getClass();
        Field stringArrayField = alt_rb.getField(
"searchJListFields" );
25        searchJListFields = (String[][] )stringArrayField.get(
alt_rb );
        } catch ( NoSuchFieldException nsfe ) {
        } catch ( IllegalAccessException iae ) {
        } // endtry

30    //
    // Got field array via reflection, find its length.
    //
    int jlistmax = searchJListFields.length;
    for ( int ii = 0; ii < jlistmax; ii++ ) {
35        model.addElement( rb.getString(
searchJListFields[ii][0] ) );
    } // endfor

    searchList = new JList( model );
    searchList.setName( "TCRManagerHelpSearchJList" );
40    searchList.setLayout( new BorderLayout() );

    // Do not allow users to select more than one item
    // in list at a time.

```

```

        searchList.setSelectionMode(
ListSelectionMode.SINGLE_SELECTION );

        searchList.setSelectedIndex( startMessageIndex );
        // set initial selection to intro
5        searchList.ensureIndexIsVisible( startMessageIndex );
        // make sure it's visible

        //
        // Keep track of current help selection so we can
        // use the forward and backward pulldown menuitems.
10        //
        windowTrail.add( new java.lang.Integer( startMessageIndex
) ); // first item is position 0

        listPane = new JScrollPane( );
        listPane.setPreferredSize( new Dimension( 200, 400 ) );
15        listPane.setMinimumSize( new Dimension( 200, 100 ) );
        listPane.setHorizontalScrollBarPolicy(
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED );
        listPane.setVerticalScrollBarPolicy(
JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED );
20        listPane.getViewport().add( searchList );
        listPane.setName( "TCRManagerHelpListJScrollPane" );

        // Position selected item at top of viewport
        // in scrollpane.
        Rectangle rc = searchList.getCellBounds(
25        startMessageIndex, startMessageIndex );
        listPane.getViewport().scrollRectToVisible( rc );

        // This inner class detect users select one of
        // product name in list then enabled OKButton
        // so that users can take next action.
30        searchList.addMouseListener( new MouseAdapter() {

            public void mouseClicked( MouseEvent e ) {
                if ( e.getClickCount() == 1 ) {

                    Object tmp = searchList.getSelectedValue();
                    int jj = searchList.getSelectedIndex();

35                    changeTextPanel( jj );
                    tcrHelp.repaint(); // Have to repaint the area.

                    //
                    // Keep track of current help selection so we
                    // can use the forward

```



```

        // and backward pulldown menuitems.
        //
        windowTrail.insertElementAt( new
5  java.lang.Integer( jj ), ++windowTrailPosition ); // item's
                                                    // position

        } // endif

    }
                                                    }); // end of
                                                    //addMouseListener

} // end method

```

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2

## APPENDIX B

```
/**
 * The <em>createTextPanel</em> method creates a text panel.
 * <p>
5  * @return void
 * (c) Copyright 2002 International Business Machines
 *      Corporation
 */
private void createTextPanel() {

10     //
    // Header area.
    //
    searchPanel = new JPanel();
    searchPanel.setLayout( new BorderLayout() );
15    searchPanel.setPreferredSize( new Dimension( 500, 300 ) );
    searchPanel.setMinimumSize( new Dimension( 100, 50 ) );
    searchPanel.setName( "TCRManagerHelpTextJPanel" );

    hdr_area = new JTextArea();

    Color clr = new Color( 255, 255, 0 );

20    hdr_area.setLayout( new BorderLayout() );
    hdr_area.setPreferredSize( new Dimension( 30, 20 ) );
    hdr_area.setMinimumSize( new Dimension( 10, 50 ) );
    hdr_area.setEditable( false );
    hdr_area.setBackground( clr );
25    hdr_area.setName( "TCRManagerHelpHeaderAreaJTextArea" );

    String sys = System.getProperty( "os.arch" );

    Font fnt = new Font( "SansSerif",
java.awt.Font.BOLD|java.awt.Font.ITALIC, 18 );

    hdr_area.setFont( fnt );

30    // This makes sure the text is scrolled to the
    // top of the pane.
    hdr_area.setCaretPosition( 0 );

    // Add the header to the north.
35    searchPanel.add( "North", hdr_area );

    //
    // Contents area.
    // Do NOT set minimum size or desire size since
```

```

// we are putting this object inside a scrollpane.
// Set it there if you want. Just make the text
// unmodifiable here.
//
5      area      = new JTextArea();

      area.setEditable( false );
      area.setLayout( new BorderLayout() );

// Establish foreground color for help text.
      clr = new Color( 0, 0, 200 );
10     area.setForeground( clr );
      area.setName( "TCRManagerHelpAreaJTextArea" );

// Set initial help content
      Font fnt1 = new Font( "Serif", java.awt.Font.BOLD, 14 );
      area.setFont( fnt1 );

15     changeTextPanel( startMessageIndex );

// This makes sure the text is scrolled to the top
// of the pane.
      area.setCaretPosition( 0 );

// Create a scroll pane and add it to the center
// of the window.
20     textPane = new JScrollPane( );
      textPane.setHorizontalScrollBarPolicy(
JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED );
      textPane.setVerticalScrollBarPolicy(
25     JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED );
      textPane.setViewPortView( area );
      textPane.setName( "TCRManagerHelpTextPaneJScrollPane" );
      searchPanel.add( "Center", textPane );

} // end method

```

## APPENDIX C

```
/**
 * The method <em>changeTextPanel</em> is invoked via
 * a response from a mouse click on the JList for the Help
5  * Dialog. This method figures out what entry was clicked
 * on and provides the essential text for the JTextArea
 * for the user.
 * <p>
 * @param
10  * <p>
 * @return void
 * (c) Copyright 2002 International Business Machines
 * Corporation
 */
15 private void changeTextPanel( int item_index ) {

    Color clr = null;
    String tmp = null;
    int str_len = 0;

    try {

        //
        // In order to change the text field, we must first
        // get the header associated with the item selected
        // by the mouse click, given by the
        // index provided on the call.
20  //
        tmp = hdr_area.getText();
        str_len = tmp.length();
        String rb_tmp_str = rb.getString(
searchJListFields[item_index][0] );
25

        //
        // For the text area header, we then need to
        // strip off any preceding blanks.
        //
        int ii=0;
35  while( (rb_tmp_str.startsWith( " ", ii )) &&
        (ii < rb_tmp_str.length()) ) {
            ii++;
        } // endwhile
        rb_tmp_str = rb_tmp_str.substring( ii,
40 rb_tmp_str.length() );

        //
        // We use substring to remove those blanks
```

```

// once we know how many there are.
//
    hdr_area.replaceRange( rb_tmp_str,
5                          0,
                          str_len );

//
// We now get the get the text of the current text
// message to get its length, and then replace it
// with the requested help selection.
10 //
    tmp = area.getText();
    str_len = tmp.length();
    area.replaceRange( rb.getString(
searchJListFields[item_index][1] ),
15                          0,
                          str_len );

//
// This makes sure the text is scrolled to the
// top of the pane.
20 //
    area.setCaretPosition( 0 );

//
// Ensure foreground color for this help text. Since
// we could have had red error text up before...!
25 //
    clr = new Color( 0, 0, 200 );
    area.setForeground( clr );

} catch ( IllegalArgumentException iae ) {
} catch ( MissingResourceException mre ) {

30 //
// Establish foreground color for ERROR text.
//
    clr = new Color( 222, 0, 0 ); // RED
    area.setForeground( clr );
35 tmp = area.getText();
    str_len = tmp.length();
    area.replaceRange( rb.getString(
"tcrmanagerhelp_rberror" ) + mre.toString(),
                          0,
40                          str_len );

    } // entry

    } // end method

```

## APPENDIX D

```
/**
 * Detect users actions. Only one OKButton can do 2
 * different tasks:
5  * First click to initialize product list, then OKButton grey
 * out and if users highlight a product name from list.
 * OKButton will be enabled to allow users to do second click
 * on OK again to query the TCR DB for test results for
 * that product
10  * <p>
 * @param ActionEvent  an event
 * <p>
 * @return void
 * (c) Copyright 2002 International Business Machines
15  *      Corporation
 */
public synchronized void actionPerformed( ActionEvent event )
{
    //
20  // If the event was a pick of Quit from the pulldown,
    // then close window
    //
    if ( event.getSource() == menuQuit ) {
        setVisible(false);
25  System.gc();
        return;
    }

    //
    // Else, if the item selected was Backward, then try to
    // go backward from current position in vector.
30  //
    } else if ( event.getSource() == menuBackward ) {

        //
        // The user wants to go backwards, get previous items
        // index from vector.
35  //
        try {
            int ii = 0;
            if ( windowTrailPosition == 0 ) {
                ii = ((Integer>windowTrail.get(
40  windowTrailPosition )).intValue();    // item's position - 1
            } else {
                ii = ((Integer>windowTrail.get( --
windowTrailPosition )).intValue();    // item's position - 1
            } // endif
        }
    }
}
```

```

        //
        // Change the selected index for the search list
        // and text panel.
        //
5       searchList.setSelectedIndex( ii );    // set initial
                                                // selection to intro
        changeTextPanel( ii );
        tcrHelp.repaint();    // Have to repaint the area.

10      } catch ( IndexOutOfBoundsException ioobe ) {
        } // endtry

        //
        // Else, if the item selected was Forward, then try to
        // go forward from current position in vector.
        //
15      } else if ( event.getSource() == menuForward ) {

        //
        // The user wants to go backwards, get previous items
        // index from vector.
        //
20      int ii=0;

        //
        // If we are already at the end of the Vector,
        // then don't change index.
        //
25      if ( windowTrailPosition == windowTrail.size() ) {
        ii = ((Integer>windowTrail.get( windowTrailPosition-1
        )).intValue();    // item's position + 1

        //
        // If we are already at the end of the Vector,
        // then don't change index.
        //
30      } else if ( (windowTrailPosition+1) >=
windowTrail.size() ) {
        ii = ((Integer>windowTrail.get( windowTrailPosition
35      )).intValue();    // item's position + 1

        //
        // Else, go to next position and get its index value.
        //
        } else {
40      windowTrailPosition++; // set to current index
                                // position of current item.

```

```

        ii = ((Integer>windowTrail.get( windowTrailPosition
)).intValue();    // item's position + 1
    } // endif

    searchList.setSelectedIndex( ii );    // set initial
                                           // selection to intro
5      changeTextPanel( ii );
    tcrHelp.repaint();    // Have to repaint the area.

    //
    // OK, then quit about.
10    //
    } else if( event.getSource() == OKButton ) {

        setVisible( false );
    //    dispose();
        System.gc();
15    return;
    } // endif

} // end method

```



## APPENDIX E

```
package com.tivoli.tcrManager.Resources;
```

```
import java.util.ListResourceBundle;
```

```

5  /*****
   *
   * <dt><b>Description:</b><dd>
   * Translatable strings for Test Case Repository TCR.
   * <p>
10  * @author Erick Griffin
   * (c) Copyright 2002 International Business Machines Corporation
   *
   *****/
public class TCRHelp extends ListResourceBundle {

15  //
   // Current (partially-illustrated) list of supported JList Search entries for HELP Panel.
   //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
   // IMPORTANT NOTE:
   //      When updating the resource bundle below with new search
20  //      strings for help texts, then this table has to be updated
   //      too. This way the method that updates the fields
   //      in the JList will automatically pull the new entry names
   //      and populate them without you making any code changes..!
   //
   //      The SEARCH TEXT if for the JList on the left of the HELP
25  //      dialog. The HELP TEXT is for the JTextArea to the right
   //      of the HELP dialog. (See TCRManagerHelp.java)
   //
   //!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
30  public static String[][] searchJListFields = new String[][] {
   // Field Description:
   //
   // the SEARCH TEXT          the HELP TEXT
   // -----
   { "tcrmanagerhelp_searchtext10", "tcrmanagerhelp_helptext10" },

35  { "tcrmanagerhelp_searchtext20", "tcrmanagerhelp_helptext20" },
   { "tcrmanagerhelp_searchtext21", "tcrmanagerhelp_helptext21" },
   { "tcrmanagerhelp_searchtext210", "tcrmanagerhelp_helptext210" },
   { "tcrmanagerhelp_searchtext2100", "tcrmanagerhelp_helptext2100" },
   { "tcrmanagerhelp_searchtext2101", "tcrmanagerhelp_helptext2101" },
40  { "tcrmanagerhelp_searchtext2102", "tcrmanagerhelp_helptext2102" },
   { "tcrmanagerhelp_searchtext211", "tcrmanagerhelp_helptext211" },
   { "tcrmanagerhelp_searchtext2110", "tcrmanagerhelp_helptext2110" },
   { "tcrmanagerhelp_searchtext22", "tcrmanagerhelp_helptext22" },
   { "tcrmanagerhelp_searchtext23", "tcrmanagerhelp_helptext23" },
45  { "tcrmanagerhelp_searchtext24", "tcrmanagerhelp_helptext24" },
   { "tcrmanagerhelp_searchtext240", "tcrmanagerhelp_helptext240" },
   { "tcrmanagerhelp_searchtext25", "tcrmanagerhelp_helptext25" },

   { "tcrmanagerhelp_searchtext30", "tcrmanagerhelp_helptext30" },
   { "tcrmanagerhelp_searchtext31", "tcrmanagerhelp_helptext31" },
50  { "tcrmanagerhelp_searchtext310", "tcrmanagerhelp_helptext310" },
   { "tcrmanagerhelp_searchtext311", "tcrmanagerhelp_helptext311" },
   { "tcrmanagerhelp_searchtext312", "tcrmanagerhelp_helptext312" },

   { "tcrmanagerhelp_searchtext40", "tcrmanagerhelp_helptext40" },

55  // similar entries for remaining fields ...

```

```
};
```

```
public Object[ ][ ] getContents( )  
{  
    return contents;  
}
```

```
static final Object[ ][ ] contents =
```

```
{  
    //*****  
    //  
    // LOCALIZE THE FOLLOWING:  
    //  
    //*****
```

```
    // -----  
    //                TCRManager  
    // -----
```

```
    //  
    // Menu Subject Headings  
    //  
    { "tcrmanagerhelp_searchtext10", "1.0 Introduction" },
```

```
    { "tcrmanagerhelp_searchtext20", "2.0 TCRManager Window" },  
    { "tcrmanagerhelp_searchtext21", " 2.1 MenuBar" },  
    { "tcrmanagerhelp_searchtext210", " 2.1.0 Select" },  
    { "tcrmanagerhelp_searchtext2100", " 2.1.0.0 Save Directory" },  
    { "tcrmanagerhelp_searchtext2101", " 2.1.0.1 Server Information" },  
    { "tcrmanagerhelp_searchtext2102", " 2.1.0.2 Quit" },  
    { "tcrmanagerhelp_searchtext211", " 2.1.1 Help" },  
    { "tcrmanagerhelp_searchtext2110", " 2.1.1.0 Help Contents" },  
    { "tcrmanagerhelp_searchtext22", " 2.2 Main Window" },  
    { "tcrmanagerhelp_searchtext23", " 2.3 Instructions" },  
    { "tcrmanagerhelp_searchtext24", " 2.4 Buttons" },  
    { "tcrmanagerhelp_searchtext240", " 2.4.0 OK" },  
    { "tcrmanagerhelp_searchtext25", " 2.5 Status Line" },
```

```
    { "tcrmanagerhelp_searchtext30", "3.0 TCRManagerHelp Window" },  
    { "tcrmanagerhelp_searchtext31", " 3.1 MenuBar" },  
    { "tcrmanagerhelp_searchtext310", " 3.1.0 Backward" },  
    { "tcrmanagerhelp_searchtext311", " 3.1.1 Forward" },  
    { "tcrmanagerhelp_searchtext312", " 3.1.2 Quit" },
```

```
    { "tcrmanagerhelp_searchtext40", "4.0 TCRQuery Window" },
```

```
    // similar entries for remaining choices ...
```

```
    //  
    // 1.0 - TCRManagerHelp Introduction Section.  
    //
```

```
    { "tcrmanagerhelp_helptext10",  
        "\n"  
        + " Welcome to the Test Case Repository (TCR) Help Dialog.\n"  
        + "\n"  
        + " This help dialog is provided to afford the user some help while\n"  
        + " using the TCR Manager to filter and perform queries against\n"  
        + " the selected TCR DB.\n"  
        + "\n"  
        + " This application offers the flexibility for users to query a\n"  
        + " selected TCR DB while testcase results are still being reported.\n"  
        + " Therefore, the user can see the results of their test execution\n"  
        + " in real time. They can even perform multiple queries at once,\n"  
        + " enabling them to monitor more than one test execution.\n"
```

```

    },

//
// 2.0 - TCRManager Help.
//
5   { "tcrmanagerhelp_helpertext20",
        "\n"
        + " The TCR Manager main window will display the currently available\n"
        + " test components for the particular Test Case Repository on the\n"
10    + " selected server. The user can select a component to query results\n"
        + " from using the query dialog. Please see the appropriate help\n"
        + " for tailoring a query against a component."
    },
    { "tcrmanagerhelp_helpertext21",
        "\n"
15    + " The MenuBar is provided for user to have some way of tailoring\n"
        + " the current session properties or look and feel. Please see\n"
        + " subsections in this help for details on each.\n"
    },

// similar entries, providing associated help text, for other choices ...
20 // and some error handling, just in case

    { "tcrmanagerhelp_rerror", "\n"
        + " Help Text Missing! Resource Bundle ERROR!\n"
        + "\n"
        + " Please report this tool error to owner.\n"
25    + " The following Resource Bundle entry was NOT found.\n\n"
    },

//*****
//
// END OF MATERIAL TO LOCALIZE
//
30 //*****

};

} // end TCRHelp

```